

RRRRRRRRRRRR	TTTTTTTTTTTTTT	PPPPPPPPPPPP	AAAAAAAAAA	DDDDDDDDDDDD			
RRRRRRRRRRRR	TTTTTTTTTTTTTT	PPPPPPPPPPPP	AAAAAAAAAA	DDDDDDDDDDDD			
RRRRRRRRRRRR	TTTTTTTTTTTTTT	PPPPPPPPPPPP	AAAAAAAAAA	DDDDDDDDDDDD			
RRR	RRR	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RRR	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RRR	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RRR	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RRR	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RRR	PPP	PPP	AAA	AAA	DDD	DDD
RRRRRRRRRRRR	TTTT	PPPPPPPPPPPP	AAA	AAA	AAA	DDD	DDD
RRRRRRRRRRRR	TTTT	PPPPPPPPPPPP	AAA	AAA	AAA	DDD	DDD
RRRRRRRRRRRR	TTTT	PPPPPPPPPPPP	AAA	AAA	AAA	DDD	DDD
RRR	RRR	PPP	AAAAAAAAAAAAAAAA	DDD	DDD		
RRR	RRR	PPP	AAAAAAAAAAAAAAAA	DDD	DDD		
RRR	RRR	PPP	AAAAAAAAAAAAAAAA	DDD	DDD		
RRR	RRR	PPP	AAA	AAA	DDD	DDD	
RRR	RRR	PPP	AAA	AAA	DDD	DDD	
RRR	RRR	PPP	AAA	AAA	DDD	DDD	
RRR	RRR	PPP	AAA	AAA	DDD	DDD	
RRR	RRR	PPP	AAA	AAA	DDDDDDDDDDDD	DDDDDDDDDDDD	
RRR	RRR	PPP	AAA	AAA	DDDDDDDDDDDD	DDDDDDDDDDDD	
RRR	RRR	PPP	AAA	AAA	DDDDDDDDDDDD	DDDDDDDDDDDD	

```

CCCCCCCCC TTTTTTTTTT SSSSSSSS EEEEEEEEEEE NN NN SSSSSSSS EEEEEEEEEEE RRRRRRRR TTTTTTTTTT
CCCCCCCCC TTTTTTTTTT SSSSSSSS EEEEEEEEEEE NN NN SSSSSSSS EEEEEEEEEEE RRRRRRRR TTTTTTTTTT
CC          TT          SS          EE          NN NN SS          EE          RR          RR          TT
CC          TT          SS          EE          NN NN SS          EE          RR          RR          TT
CC          TT          SS          EE          NN NN SS          EE          RR          RR          TT
CC          TT          SS          EE          NN NN SS          EE          RR          RR          TT
CC          TT          SS          EE          NN NN SS          EE          RR          RR          TT
CC          TT          SS          EE          NN NN SS          EE          RR          RR          TT
CC          TT          SS          EE          NN NN SS          EE          RR          RR          TT
CC          TT          SS          EE          NN NN SS          EE          RR          RR          TT
CC          TT          SS          EE          NN NN SS          EE          RR          RR          TT
CCCCCCCCC TT          SSSSSSSS EEEEEEEEEEE NN NN SSSSSSSS EEEEEEEEEEE RRRRRRRR TT          ....
CCCCCCCCC TT          SSSSSSSS EEEEEEEEEEE NN NN SSSSSSSS EEEEEEEEEEE RRRRRRRR TT          ....
                                     ....
                                     ....

LLLLLLLLLLL IIIIIIII SSSSSSSS
LLLLLLLLLLL IIIIIIII SSSSSSSS
LLLL          II          SS
LLLL          II          SS
LLLL          II          SS
LLLL          II          SS
LLLL          II          SSSSSS
LLLL          II          SSSSSS
LLLL          II          SS
LLLL          II          SS
LLLL          II          SS
LLLL          II          SS
LLLLLLLLLLL IIIIIIII SSSSSSSS
LLLLLLLLLLL IIIIIIII SSSSSSSS

```


(2) 57
(2) 217

DECLARATIONS
CTSENSECHAR - Map READ_CHAR into CHAR using VMS data

```

0000 1 .TITLE CTSENSERT - RTPAD/CTERM SENSE CHARACTERISTICS
0000 2 .IDENT 'V04-000'
0000 3 .ENABLE SUPPRESSION
0000 4 .DISABLE GLOBAL
0000 5 :
0000 6 :*****
0000 7 :*
0000 8 :* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 9 :* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 10 :* ALL RIGHTS RESERVED.
0000 11 :*
0000 12 :* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 13 :* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 14 :* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 15 :* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 16 :* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 17 :* TRANSFERRED.
0000 18 :*
0000 19 :* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 20 :* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 21 :* CORPORATION.
0000 22 :*
0000 23 :* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 24 :* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 25 :*
0000 26 :*
0000 27 :*****
0000 28 :
0000 29 :
0000 30 :++
0000 31 :
0000 32 : FACILITY:
0000 33 :
0000 34 : CTERM Remote terminal protocol driver
0000 35 :
0000 36 : ABSTRACT:
0000 37 :
0000 38 : This module is called to map a characteristics item list received
0000 39 : from the net into VMS terminal characteristics.
0000 40 :
0000 41 : ENVIRONMENT:
0000 42 :
0000 43 :
0000 44 :
0000 45 :--
0000 46 :
0000 47 : AUTHOR: Jake VanNoy, CREATION DATE: 23-Aug-1982
0000 48 :
0000 49 : MODIFIED BY:
0000 50 :
0000 51 : V03-001 JLV0297 Jake VanNoy 28-JUL-1983
0000 52 : Change characteristics symbols.
0000 53 :
0000 54 : **
0000 55 :

```

CT
Sy

AU
AU
CH
CH
CH
CH
CH
CH
CH
CO
CR
CT
CT
CT
CT
CT
CT
ER
FL
FO
GE
HO
IG
IG
IN
IN
IN
IN
LF
LI
LO
LO
MA
MO
MO
NO
OU
OU
OU
OU
PA
PA
PA
PH
PO
RA
SE
SE
SE
ST
SW
SW
TE
TE
TE
TT
TT


```
0000 57      .SBTTL  DECLARATIONS
0000 58      :
0000 59      : INCLUDE FILES:
0000 60      :
0000 61      $DCDEF
0000 62      $SSDEF
0000 63      $TTDEF
0000 64      $TT2DEF
0000 65      $TSADEF
0000 66
0000 67      :
0000 68      : MACROS:
0000 69      :
0000 70
0000 71      .MACRO  CHAR      LABEL
0000 72      .WORD
0000 73      .WORD      char_type!char_count
0000 74      char_count = char_count + 1
0000 75      .ENDM  CHAR
0000 76
0000 77      .MACRO  BAUDTAB BPS
0000 78      .WORD  BPS
0000 79      .ENDM  BAUDTAB
0000 80
0000 81      .MACRO  TERMTAB TERM,VALUE
0000 82      .BYTE  VALUE
0000 83      .ASCII /TERM/
0000 84      .ENDM  TERMTAB
0000 85
0000 86      .MACRO  SENSETAB CHAR
0000 87      .WORD
0000 88      .ENDM  SENSETAB      char_type!char
0000 89
0000 90
0000 91      :
0000 92      : EQUATED SYMBOLS:
0000 93      :
0000 94
00000000 0000 95 physical = ch$c_physicala8      : 0 leftshifted 1 byte
00000100 0000 96 logical   = ch$c_logicala8    : 1 leftshifted 1 byte
00000200 0000 97 cterm    = ch$c_cterma8      : 2 leftshifted 1 byte
0000 98
00000000 0000 99 TT_BUF    = 0
00000004 0000 100 TT_CHAR1  = 4
00000008 0000 101 TT_CHAR2  = 8
00000000 0000 102 TT_IOSB   = 0
0000 103
0000 104      :
0000 105      : OWN STORAGE:
0000 106      :
0000 107
00000000 0000 108      .PSECT RTPAD,NOWRT
0000 109
0000 110 SENSE_TABLE:
0000 111
00000000 0000 112 char_type = physical
00000001 0000 113 char_count = 1
```

```
0000 114 CHAR INPUT_SPEED : 1
0004 115 CHAR OUTPUT_SPEED : 2
0008 116 CHAR CHARACTER_SIZE : 3
000C 117 CHAR PARITY_ENABLE : 4
0010 118 CHAR PARITY_TYPE : 5
0014 119 CHAR MODEM_PRESENT : 6
0018 120 CHAR AUTOBAUD_DETECT : 7
001C 121 CHAR MANAGEMENT_GUARANTEED : 8
0020 122 CHAR SWITCH_CHAR_1 : 9
0024 123 CHAR SWITCH_CHAR_2 : 10
0028 124
00000100 0028 125 char_type = logical
00000001 0028 126 char_count = 1
0028 127
0028 128 CHAR MODE_WRITING_ALLOWED : 1
002C 129 CHAR TERMINAL_TYPE : 2
0030 130 CHAR TERMINAL_SUBTYPE : 3
0034 131 CHAR OUTPUT_FLOW_CONTROL : 4
0038 132 CHAR OUTPUT_PAGE_STOP : 5
003C 133 CHAR FLOW_CHAR_PASS_THRU : 6
0040 134 CHAR INPUT_FLOW_CONTROL : 7
0044 135 CHAR LOSS_NOTIFICATION : 8
0048 136 CHAR LINE_WIDTH : 9
004C 137 CHAR PAGE_LENGTH : 10
0050 138 CHAR STOP_LENGTH : 11
0054 139 CHAR CR_FILL : 12
0058 140 CHAR LF_FILL : 13
005C 141 CHAR WRAP : 14
0060 142 CHAR HORIZONTAL_TAB : 15
0064 143 CHAR VERTICAL_TAB : 16
0068 144 CHAR FORM_FEED : 17
006C 145
00000200 006C 146 char_type = cterm
00000001 006C 147 char_count = 1
006C 148
006C 149 CHAR IGNORE_INPUT : 1
0070 150 CHAR CHAR_ATTRIBUTES : 2
0074 151 CHAR CONTROL_O_PASS_THRU : 3
0078 152 CHAR RAISE_INPUT : 4
007C 153 CHAR NORMAL_ECHO : 5
0080 154 ;*** CHAR REQUEST_PROCESSING : 6
0080 155 CHAR INPUT_ESCAPE_ENABLE : 7
0084 156 CHAR OUTPUT_ESCAPE_ENABLE : 8
0088 157 CHAR INPUT_COUNT_STATE : 9
008C 158 CHAR AUTO_PROMPT : 10
0090 159 CHAR ERROR_PROCESSING : 11
0094 160
FFFF 0094 161 .WORD -1 : End of list (any negative number)
0096 162
0096 163 SENSE_BAUD: : Table must be in ascending order
0096 164 BAUDTAB 0 : 0
0098 165 BAUDTAB 50 : 1
009A 166 BAUDTAB 75 : 2
009C 167 BAUDTAB 110 : 3
009E 168 BAUDTAB 134 : 4
00A0 169 BAUDTAB 150 : 5
00A2 170 BAUDTAB 300 : 6
```


00A4	171	BAUDTAB	600	:	7
00A6	172	BAUDTAB	1200	:	8
00A8	173	BAUDTAB	1800	:	9
00AA	174	BAUDTAB	2000	:	10
00AC	175	BAUDTAB	2400	:	11
00AE	176	BAUDTAB	3600	:	12
00B0	177	BAUDTAB	4800	:	13
00B2	178	BAUDTAB	7200	:	14
00B4	179	BAUDTAB	9600	:	15
00B6	180	BAUDTAB	19200	:	16
00B8	181				

00B8	182	TERM_TABLE:			
00B8	183	TERMTAB	VT100,	TTS_VT100	
00BF	184	TERMTAB	VT101,	TTS_VT101	
00C6	185	TERMTAB	VT102,	TTS_VT102	
00CD	186	TERMTAB	VT105,	TTS_VT105	
00D4	187	TERMTAB	VT125,	TTS_VT125	
00DB	188	TERMTAB	VT131,	TTS_VT131	
00E2	189	TERMTAB	VT132,	TTS_VT132	
00E9	190	TERMTAB	VT52,	TTS_VT52	
00EF	191	TERMTAB	VT05,	TTS_VT05	
00F5	192	TERMTAB	VK100,	TTS_VK100	
00FC	193	TERMTAB	VT173,	TTS_VT173	
0103	194	:	TERMTAB	TTS_FT1	
0103	195	:	TERMTAB	TTS_FT2	
0103	196	:	TERMTAB	TTS_FT3	
0103	197	:	TERMTAB	TTS_FT4	
0103	198	:	TERMTAB	TTS_FT5	
0103	199	:	TERMTAB	TTS_FT6	
0103	200	:	TERMTAB	TTS_FT7	
0103	201	:	TERMTAB	TTS_FT8	
0103	202		TERMTAB	LA36,	TTS_LA36
0109	203		TERMTAB	LA12,	TTS_LA12
010F	204		TERMTAB	LA34,	TTS_LA34
0115	205		TERMTAB	LA38,	TTS_LA38
011B	206		TERMTAB	LA12,	TTS_LA12
0121	207		TERMTAB	LA100,	TTS_LA100
0128	208		TERMTAB	LA24,	TTS_LA24
012E	209		TERMTAB	LQP02,	TTS_LQP02
0135	210		TERMTAB	VT55,	TTS_VT55
00	013B	211	.BYTE	0	

4E 57 4F 4E 4B 4E 55 00' 07

013C	212				
013C	213	UNKNOWN_TT:	.ascii	/UNKNOWN/	

0144	214				
0144	215	.nlist	meb		
0144	216				
0144	217	.SBTTL	CTSENSECHAR - Map READ_CHAR into CHAR using VMS data		
0144	218				
0144	219	:			
0144	220	:	Inputs:		
0144	221	:	R0 - CTP		
0144	222	:	R2 - address of requested characteristics buffer (READ_CHAR)		
0144	223	:	R3 - address of buffer to be written		
0144	224	:	R9 - address of 12 buffer of current characteristics		
0144	225	:	R10 - address of iosb from operation		
0144	226	:			

; Table is in order for most likely first

```
0144 227 : Outputs:
0144 228 : R1 is buffer length from start
0144 229 : buffer pointed to by R3 is complete
0144 230 : All registers destroyed
0144 231 :
0144 232 CTSENSECHAR::
0144 233
51 28 53 DD 0144 234 PUSHL R3 ; Save
50 2A A0 3C 0146 235 MOVZWL CTP$W_MSGSIZE(R0),R1 ; message size
5B 51 50 9E 014A 236 MOVAB CTP$B_MSGTYPE(R0),R0 ; base address
C1 014E 237 ADDL3 R0,R1,R11 ; add together for end of message address
0152 238
0152 239 10$:
56 82 3C 0152 240 MOVZWL (R2)+,R6 ; Get parameter
5B 52 D1 0155 241 CMPL R2,R11 ; past end? *** should be one up?
11 1A 0158 242 BGTRU 25$ ; Done, exit
83 56 B0 015A 243 MOVW R6,(R3)+ ; Move into table
57 FE9F CF 9E 015D 244 MOVAB SENSE_TABLE,R7 ; Get table address
0162 245 20$:
87 56 B1 0162 246 CMPW R6,(R7)+ ; Compare to table entry
07 13 0165 247 BEQL 30$ ; Branch if match found
87 B5 0167 248 TSTW (R7)+ ; Advance pointer past routine
F7 18 0169 249 BGEQ 20$ ; Loop if greater than or equal to zero
016B 250
016B 251 ; PANIC ; 'invalid sense mode returned'
016B 252 25$:
0193 31 016B 253 BRW POST_SENSE_EXIT ; exit code
016E 254
016E 255 :
016E 256 : Dispatch to routine
016E 257 :
016E 258 30$:
56 0000017F'EF 9E 016E 259 MOVAB SENSE_ROUTINES,R6 ; Base
57 67 3C 0175 260 MOVZWL (R7),R7 ; get routine address
57 56 C0 0178 261 ADDL2 R6,R7 ; Add offset to base for routine address
67 16 017B 262 JSB (R7) ; jsb to routine
D3 11 017D 263 BRB 10$ ; Do next parameter
017F 264
017F 265 :
017F 266 : All these routines have R0,R1,R4,R6-R8 as scratch
017F 267 :
017F 268 :
```



```
017F 270 SENSE_ROUTINES:
017F 271
017F 272 ; physical
017F 273
50 03 AA 9A 017F 274 INPUT_SPEED: ; physical 1
06 13 017F 275 MOVZBL TT_IOSB+3(R10),R0 ; Get speed
83 50 0E 10 0183 276 BEQL OUTPUT_SPEED ; If zero, then same as output speed
05 B0 0185 277 BSBB GET_SPEED ; map from TT$C_BAUD_XXXX
05 05 0187 278 MOVW R0,(R3)+ ; set receive speed
018A 279 RSB ; Return
018B 280
50 02 AA 9A 018B 281 OUTPUT_SPEED: ; physical 2
04 10 018B 282 MOVZBL TT_IOSB+2(R10),R0 ; set transmit speed
83 50 B0 018F 283 BSBB GET_SPEED ; map from TT$C_BAUD_XXXX
05 05 0191 284 MOVW R0,(R3)+ ; set transmit speed
0194 285 RSB ; Return
0195 286
0195 287 ; Local routine to map TT$C_BAUD rates into speed
0195 288
0195 289 ; Input:
0195 290 R0 - Baud rate in TT$C_BAUD_XXX terms
0195 291
0195 292 ; Output:
0195 293 R0 - baud rate in BPS
0195 294
51 FEFD CF 9E 0195 295 GET_SPEED:
50 6140 3C 0195 296 MOVAB SENSE_BAUD,R1 ; Get table
05 05 019A 297 MOVZWL (R1)[R0],R0 ; Index in and fetch value
019F 298 RSB ; Return
019F 300 CHARACTER_SIZE: ; physical 3
50 07 90 019F 301 MOVB #7,R0 ; Character size
03 04 A9 E1 01A2 302 BBC #TT$V_EIGHTBIT,- ; check characteristic
50 08 90 01A4 303 TT_CHAR1(R9),10$ ; Character size
83 50 9B 01A7 304 MOVB #8,R0
01AA 305
83 50 9B 01AA 306 10$: MOVZBW R0,(R3)+ ; Set value
05 05 01AD 307 RSB ; Return
01AE 308
01AE 309 PARITY_ENABLE: ; physical 4
50 50 94 01AE 310 CLRB R0 ; Set noparity
06 06 E1 01B0 311 BBC #TT$V_PARITY,- ; Check parity
03 06 AA 90 01B2 312 TT_IOSB+6(R10),10$ ; Set parity enabled
50 01 90 01B5 313 MOVB #1,R0
83 50 90 01B8 314 10$: MOVB R0,(R3)+
05 05 01BB 315 MOVB R0,(R3)+ ; Return
01BC 316
01BC 317 PARITY_TYPE: ; physical 5
50 01 90 01BC 318 MOVB #1,R0 ; Assume even
07 07 E1 01BF 319 BBC #TT$V_ODD,-
03 06 AA 90 01C1 320 TT_IOSB+6(R10),10$ ; branch if even
50 02 90 01C4 321 MOVB #2,R0 ; Set odd
83 50 9B 01C7 322 10$: MOVZBW R0,(R3)+ ; Set parity
05 05 01CA 323 MOVZBW R0,(R3)+ ; Return
01CB 324
01CB 325
01CB 326
```

```
03 04 A9 50 94 01CB 327 MODEM_PRESENT: ; physical 6
15 E1 01CB 328 CLR B0 ; Get Boolean
01 90 01CD 329 BBC #TT$V MODEM, -
83 50 90 01D2 330 TT_CHAR1(R9),10$ ; check modem bit
01D5 331 MOV B #1,R0
01D5 332 10$:
05 01D8 333 MOV B R0,(R3)+ ; Set value
01D9 334 RSB ; Return
01D9 335
50 94 01D9 336 AUTOBAUD_DETECT: ; physical 7
01 E1 01DB 337 CLR B0
03 08 A9 90 01DD 338 BBC #TT2$V AUTOBAUD, -
50 01 90 01E0 339 TT_CHAR2(R9),10$ ; Set autobaud
83 50 90 01E3 340 MOV B #1,R0
01E3 341 10$:
05 01E6 342 MOV B R0,(R3)+
01E7 343 RSB ; Return
01E7 344
83 94 01E7 345 MANAGEMENT_GUARANTEED: ; physical 8
05 01E9 346 CLR B (R3)+ ; clear
01EA 347 10$: RSB ; Return
01EA 348
0111 30 01EA 349 SWITCH_CHAR_1: ; physical 9
05 01EA 350 BSBW IGNORE_STRING ; Ignore this string data
01ED 351 RSB ; Return
01EE 352
010D 30 01EE 353 SWITCH_CHAR_2: ; physical 10
05 01EE 354 BSBW IGNORE_STRING ; Ignore this string data
01F1 355 RSB ; Return
01F2 356
```



```
01F2 358 ; logical
01F2 359
01F2 360 MODE_WRITING_ALLOWED: ; logical 1
83 94 01F2 361 CLR B (R3)+
05 01F4 362 10$: RSB ; Return
01F5 363
01F5 364 TERMINAL_TYPE: ; logical 2
01F5 365
01F5 366 ; Bits are:
01F5 367 : 0 - known/unknown
01F5 368 : 1 - scope/hardcopy
01F5 369 :
50 D4 01F5 370 CLR L R0
01F7 371 ASSUME TT$ UNKNOWN EQ 0 ; assume unknown = 0
01 01 A9 95 01F7 372 TST B TT BUF+1(R9) ; is it unknown?
03 03 13 01FA 373 BEQ L 10$ ; Branch if unknown
50 01 88 01FC 374 BIS B #CTPSM_CH_KNOWN,R0 ; Set known bit
01FF 375 10$:
01FF 376 BBC #TT$V SCOPE,- ; check for scope
03 04 A9 0201 377 TT CHAR1(R9),20$ ; set scope
50 02 88 0204 378 BIS B #CTPSM_CH_SCOPE,R0
0207 379 20$:
83 50 9B 0207 380 MOVZBW R0,(R3)+ ; set value
05 020A 381 RSB ; Return
020B 382
020B 383 TERMINAL_SUBTYPE: ; logical 3
020B 384
50 01 A9 9A 020B 385 MOVZBL TT BUF+1(R9),R0 ; Fetch terminal type byte
51 FEAS CF 9E 020F 386 MOVAB TERM_TABLE,R1 ; Address of table
0214 387 20$:
54 81 9A 0214 388 MOVZBL (R1)+,R4 ; Get terminal type from table
0D 13 0217 389 BEQ L 30$ ; if zero, end of list => unknown
54 50 91 0219 390 CMPB R0,R4 ; Compare terminal types
0D 13 021C 391 BEQ L 40$ ; Match, branch
54 81 9A 021E 392 MOVZBL (R1)+,R4 ; get length of string
51 54 C0 0221 393 ADDL R4,R1 ; Add to pointer
EE 11 0224 394 BRB 20$ ; Loop
0226 395 30$:
51 FF12 CF 9E 0226 396 MOVAB UNKNOWN_TT,R1 ; 'unknown'
022B 397 40$:
54 34 BB 022B 398 PUSH R #^M<R2,R4,R5> ; Save registers
54 61 9A 022D 399 MOVZBL (R1),R4 ; Get length
54 54 D6 0230 400 INCL R4 ; Include byte for count
63 61 54 28 0232 401 MOV C3 R4,(R1),(R3) ; Move to buffer, update R3
34 BA 0236 402 POP R #^M<R2,R4,R5> ; restore
0238 403
05 0238 404 RSB ; Return
0239 405
0239 406 OUTPUT_FLOW_CONTROL: ; logical 4
50 94 0239 407 CLR B R0 ; assume no
05 01 E1 023B 408 BBC #TT$V TTSYNC,- ; branch if not tt sync
03 04 A9 023D 409 TT CHAR1(R9),10$ ; Set true
50 01 90 0240 410 MOV B #1,R0
83 50 90 0243 411 10$:
05 0243 412 MOV B R0,(R3)+ ; Set data
0246 413 RSB ; Return
0247 414
```

```
0247 415 OUTPUT_PAGE STOP: ; logical 5
50 94 0247 416 CLR B R0 ; assume no
03 04 0E E1 0249 417 BBC #TT$V HOLDSscreen,-
50 01 90 024B 418 TT_CHAR1(R9),10$ ; branch if not holdscreen
83 50 90 024E 419 MOV B #1,R0 ; Set true
0251 420 10$:
05 90 0251 421 MOV B R0,(R3)+ ; Set data
0254 422 RSB ; Return
0255 423
0255 424 FLOW_CHAR_PASS_THRU: ; logical 6
83 94 0255 425 CLR B (R3)+
05 0257 426 10$: RSB ; Return
0258 427
0258 428 INPUT_FLOW CONTROL: ; logical 7
50 94 0258 429 CLR B R0 ; assume no
04 0E E1 025A 430 BBC #TT$V HOSTSYNc,-
03 04 0A A9 025C 431 TT_CHAR1(R9),10$ ; branch if not host sync
50 01 90 025F 432 MOV B #1,R0 ; Set true
83 50 90 0262 433 10$:
05 90 0262 434 MOV B R0,(R3)+ ; Set data
0265 435 RSB ; Return
0266 436
0266 437 LOSS_NOTIFICATION: ; logical 8
83 94 0266 438 CLR B (R3)+ ; ignore
05 0268 439 10$: RSB ; Bell on loss data?
0269 441
0269 442 LINE_WIDTH: ; logical 9
83 02 A9 B0 0269 443 MOV W TT_BUF+2(R9),(R3)+ ; Line width
05 026D 444 RSB ; Return
026E 445
026E 446 PAGE_LENGTH: ; logical 10
83 07 A9 9B 026E 447 MOV ZBW TT_CHAR1+3(R9),(R3)+ ; Page length
05 0272 448 RSB ; Return
0273 449
0273 450 STOP_LENGTH: ; logical 11
83 B4 0273 451 CLR W (R3)+ ; ** ignore
05 0275 452 RSB ; Return
0276 453
0276 454 CR_FILL: ; logical 12
50 B4 0276 455 CLR W R0 ; Assume no cr fill
04 0A E1 0278 456 BBC #TT$V CRFILL,-
50 04 0A A9 027A 457 TT_CHAR1(R9),10$ ; branch if off
50 06 AA B0 027D 458 MOV W TT_IOSB+6(R10),R0 ; set on
83 50 B0 0281 459 10$:
05 90 0281 460 MOV W R0,(R3)+
0284 461 RSB ; Return
0285 462
0285 463 LF_FILL: ; logical 13
50 B4 0285 464 CLR W R0 ; Assume no lf fill
04 0B E1 0287 465 BBC #TT$V LFFILL,-
50 04 0A A9 0289 466 TT_CHAR1(R9),10$ ; branch if off
50 07 AA B0 028C 467 MOV W TT_IOSB+7(R10),R0 ; set on
83 50 B0 0290 468 10$:
05 90 0290 469 MOV W R0,(R3)+
0293 470 RSB ; Return
0294 471
```



```
50 01 B0 0294 472 WRAP: ; logical 14
03 04 09 E1 0294 473 MOVW #1,R0 ; assume no wrap
03 04 A9 0297 474 BBC #TT$V WRAP,-
50 04 B0 0299 475 TT_CHAR1(R9),10$ ; branch if no wrap
03 04 04 B0 029C 476 MOVW #4,R0 ; Set wrap
83 50 B0 029F 477 10$: ;
03 04 05 B0 029F 478 MOVW R0,(R3)+ ; Set data
03 04 05 B0 02A2 479 RSB ; Return
03 04 05 B0 02A3 480 ;
03 04 05 B0 02A3 481 HORIZONTAL TAB: ; logical 15
03 04 08 E1 02A3 482 MOVW #2,R0 ; Assume no mechtabs
03 04 A9 02A6 483 BBC #TT$V MECHTAB,-
50 01 B0 02A8 484 TT_CHAR1(R9),10$ ; Set mechtabs
50 01 B0 02AB 485 MOVW #1,R0 ; set mechtabs
83 50 B0 02AE 486 10$: ;
03 04 05 B0 02AE 487 MOVW R0,(R3)+ ; set data
03 04 05 B0 02B1 488 RSB ; Return
03 04 05 B0 02B2 489 ;
03 04 83 B4 02B2 490 VERTICAL TAB: ; logical 16
03 04 05 B4 02B2 491 CLRW (R3)+ ; ignore
03 04 05 B4 02B4 492 RSB ; Return
03 04 05 B4 02B5 493 ;
03 04 05 B0 02B5 494 FORM_FEED: ; logical 17
03 04 13 E1 02B5 495 MOVW #2,R0 ; Assume no mechform
03 04 A9 02B8 496 BBC #TT$V MECHFORM,-
50 01 B0 02BA 497 TT_CHAR1(R9),10$ ; Set mechform
50 01 B0 02BD 498 MOVW #1,R0 ; set mechform
83 50 B0 02C0 499 10$: ;
03 04 05 B0 02C0 500 MOVW R0,(R3)+ ; set data
03 04 05 B0 02C3 501 RSB ; Return
03 04 05 B0 02C4 502
03 04 05 B0 02C4 503
```

```
02C4 505 ; cterm
02C4 506
02C4 507 IGNORE_INPUT: ; cterm 1
83 94 02C4 508 CLR B (R3)+ ; Ignore
05 02C6 509 10$: RSB ; Return
02C7 510
02C7 511 CHAR_ATTRIBUTES: ; cterm 2
02C7 512
02C7 513
05 02C7 514 RSB
02C8 515
02C8 516 CONTROL_O_PASS_THRU: ; cterm 3
83 94 02C8 517 CLR B (R3)+ ; Ignore
05 02CA 518 10$: RSB ; Return
02CB 519
02CB 520 RAISE_INPUT: ; cterm 4
50 94 02CB 521 CLR B R0 ; Assume lower case (i.e. NO RAISE)
07 E0 02CD 522 BBS #TT$V_LOWER,-
03 04 A9 02CF 523 TT_CHAR1(R9),10$ ; Branch if NO RAISE
50 01 90 02D2 524 MOV B #1,R0 ; Set RAISE
02D5 525 10$:
83 50 90 02D5 526 MOV B R0,(R3)+ ; set data
05 02D8 527 RSB ; Return
02D9 528
02D9 529 NORMAL_ECHO: ; cterm 5
50 94 02D9 530 CLR B R0 ; (note opposite sense)
01 E0 02DB 531 BBS #TT$V_NOECHO,-
03 04 A9 02DD 532 TT_CHAR1(R9),10$ ; check noecho
50 01 90 02E0 533 MOV B #1,R0
02E3 534 10$:
83 50 90 02E3 535 MOV B R0,(R3)+ ; set data
05 02E6 536 RSB ; Return
02E7 537
02E7 538 ;*** REQUEST_PROCESSING: ; cterm 6
02E7 539
02E7 540 INPUT_ESCAPE_ENABLE: ; cterm 7
02E7 541 OUTPUT_ESCAPE_ENABLE: ; cterm 8
50 94 02E7 542 CLR B R0
03 E1 02E9 543 BBC #TT$V_ESCAPE,-
03 04 A9 02EB 544 TT_CHAR1(R9),10$
50 01 90 02EE 545 MOV B #1,R0 ; Set escape
02F1 546 10$:
83 50 90 02F1 547 MOV B R0,(R3)+
05 02F4 548 RSB ; Return
02F5 549
02F5 550 INPUT_COUNT_STATE: ; cterm 9
83 B4 02F5 551 CLR B (R3)+ ; Ignore
05 02F7 552 RSB
02F8 553
02F8 554 AUTO_PROMPT: ; cterm 10
83 94 02F8 555 CLR B (R3)+ ; Ignore
05 02FA 556 10$: RSB ; Return
02FB 557
02FB 558 ERROR_PROCESSING: ; cterm 11
83 94 02FB 559 CLR B (R3)+ ; Ignore
05 02FD 560 10$: RSB ; Return
02FE 561
```


CTSENSERT
V04-000

- RTPAD/CTERM SENSE CHARACTERISTICS D 12
CTSENSECHAR - Map READ_CHAR into CHAR us 16-SEP-1984 02:11:02 VAX/VMS Macro V04-00 Page 12
5-SEP-1984 03:14:55 [RTPAD.SRC]CTSENSERT.MAR;1 (5)

83	94	02FE	562	IGNORE_STRING: CLRB RSB	(R3)+
	05	02FE	563		
		02FE	564		
		0300	565		
		0301	566		

```

0301 568 POST_SENSE_EXIT:
0301 569
0301 570
0301 571 ; Return R1 as correct length of characteristics
0301 572 ;
0301 573 ;
51 53 51 8ED0 0301 573 POPL R1 ; start of buffer
51 C3 0304 574 SUBL3 R1,R3,R1 ; R1 = R3 - R1
05 0308 575 RSB
0309 576
0309 577 .end

```


CTSENSERT
Symbol table

- RTPAD/CTERM SENSE CHARACTERISTICS F 12

16-SEP-1984 02:11:02 VAX/VMS Macro V04-00
5-SEP-1984 03:14:55 [RTPAD.SRC]CTSENSERT.MAR;1

Page 14
(6)

AUTOBAUD_DETECT	000001D9	R	02	TT\$V_ESCAPE	= 00000003		
AUTO_PROMPT	000002F8	R	02	TT\$V_HOLDSCREEN	= 0000000E		
CH\$C_CTERM	= 00000002			TT\$V_HOSTSYNC	= 00000004		
CH\$C_LOGICAL	= 00000001			TT\$V_LFFILL	= 0000000B		
CH\$C_PHYSICAL	= 00000000			TT\$V_LOWER	= 00000007		
CHARACTER_SIZE	0000019F	R	02	TT\$V_MECHFORM	= 00000013		
CHAR_ATTRIBUTES	000002C7	R	02	TT\$V_MECHTAB	= 00000008		
CHAR_COUNT	= 0000000B			TT\$V_MODEM	= 00000015		
CHAR_TYPE	= 00000200			TT\$V_NOECHO	= 00000001		
CONTROL_O_PASS_THRU	000002C8	R	02	TT\$V_ODD	= 00000007		
CR_FILL	00000276	R	02	TT\$V_PARITY	= 00000006		
CTERM	= 00000200			TT\$V_SCOPE	= 0000000C		
CTPSB_MSGTYPE	= 0000002A			TT\$V_TTSYNC	= 00000005		
CTPSM_CH_KNOWN	= 00000001			TT\$V_WRAP	= 00000009		
CTPSM_CH_SCOPE	= 00000002			TT\$_CA100	= 00000025		
CTPSW_MSGSIZE	= 00000028			TT\$_LA12	= 00000024		
CTSENSECHAR	00000144	RG	02	TT\$_LA24	= 00000025		
ERROR_PROCESSING	000002FB	R	02	TT\$_LA34	= 00000022		
FLOW_CHAR_PASS_THRU	00000255	R	02	TT\$_LA36	= 00000020		
FORM_FEED	000002B5	R	02	TT\$_LA38	= 00000023		
GET_SPEED	00000195	R	02	TT\$_LQP02	= 00000026		
HORIZONTAL_TAB	000002A3	R	02	TT\$_UNKNOWN	= 00000000		
IGNORE_INPUT	000002C4	R	02	TT\$_VK100	= 00000002		
IGNORE_STRING	000002FE	R	02	TT\$_VT05	= 00000001		
INPUT_COUNT_STATE	000002F5	R	02	TT\$_VT100	= 00000060		
INPUT_ESCAPE_ENABLE	000002E7	R	02	TT\$_VT101	= 00000061		
INPUT_FLOW_CONTROL	00000258	R	02	TT\$_VT102	= 00000062		
INPUT_SPEED	0000017F	R	02	TT\$_VT105	= 00000063		
LF_FILL	00000285	R	02	TT\$_VT125	= 00000064		
LINE_WIDTH	00000269	R	02	TT\$_VT131	= 00000065		
LOGICAL	= 00000100			TT\$_VT132	= 00000066		
LOSS_NOTIFICATION	00000266	R	02	TT\$_VT173	= 00000003		
MANAGEMENT_GUARANTEED	000001E7	R	02	TT\$_VT52	= 00000040		
MODEM_PRESENT	000001CB	R	02	TT\$_VT55	= 00000041		
MODE_WRITING_ALLOWED	000001F2	R	02	TT2\$V_AUTOBAUD	= 00000001		
NORMAL_ECHO	000002D9	R	02	TT_BUF	= 00000000		
OUTPUT_ESCAPE_ENABLE	000002E7	R	02	TT_CHAR1	= 00000004		
OUTPUT_FLOW_CONTROL	00000239	R	02	TT_CHAR2	= 00000008		
OUTPUT_PAGE_STOP	00000247	R	02	TT_IOSB	= 00000000		
OUTPUT_SPEED	0000018B	R	02	UNKNOWN TT	0000013C	R	02
PAGE_LENGTH	0000026E	R	02	VERTICAL_TAB	000002B2	R	02
PARITY_ENABLE	000001AE	R	02	WRAP	00000294	R	02
PARITY_TYPE	000001BC	R	02				
PHYSICAL	= 00000000						
POST_SENSE_EXIT	00000301	R	02				
RAISE_INPUT	000002CB	R	02				
SENSE_BAUD	00000096	R	02				
SENSE_ROUTINES	0000017F	R	02				
SENSE_TABLE	00000000	R	02				
STOP_LENGTH	00000273	R	02				
SWITCH_CHAR_1	000001EA	R	02				
SWITCH_CHAR_2	000001EE	R	02				
TERMINAL_SUBTYPE	0000020B	R	02				
TERMINAL_TYPE	000001F5	R	02				
TERM_TABLE	000000B8	R	02				
TT\$V_CRFILL	= 0000000A						
TT\$V_EIGHTBIT	= 0000000F						

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes														
. ABS .	00000000 (0.)	00 (0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE				
\$AB\$\$	00000000 (0.)	01 (1.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE				
RTPAD	00000309 (777.)	02 (2.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	NOWRT	NOVEC	BYTE				

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.05	00:00:01.46
Command processing	122	00:00:00.50	00:00:03.53
Pass 1	367	00:00:08.00	00:00:35.16
Symbol table sort	0	00:00:01.37	00:00:03.69
Pass 2	106	00:00:01.64	00:00:06.41
Symbol table output	13	00:00:00.07	00:00:00.07
Psect synopsis output	1	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	643	00:00:11.65	00:00:50.34

The working set limit was 1500 pages.

68631 bytes (135 pages) of virtual memory were used to buffer the intermediate code.

There were 70 pages of symbol table space allocated to hold 1274 non-local and 33 local symbols.

577 source lines were read in Pass 1, producing 16 object records in Pass 2.

17 pages of virtual memory were used to define 16 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
_\$255\$DUA28:[RTPAD.OBJ]RTPAD.MLB;1	1
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	0
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	8
TOTALS (all libraries)	9

1303 GETs were required to define 9 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:CTSENSERT/OBJ=OBJ\$:CTSENSERT MSRC\$:CTSENSERT/UPDATE=(ENH\$:CTSENSERT)+EXECML\$/LIB+LIB\$:RTPAD/LIB

0333 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

CTDSENSE
LIS

CTDUMSPEC
LIS

CTSENSE
LIS

RSTRT
LIS

CTERMRT
LIS

DTE_DF03
LIS

CTSETRT
LIS

CTOSET
LIS